Innovations in Programming Models, Compilers, and Runtime Systems for Dynamic Adaptive Event-Driven Execution Models

Rishi Khan
rishi@etinternational.com
ET International, Inc.

Benoit Meister
meister@reservoir.com
Reservoir Labs

David Padua
padua@illinois.edu
UIUC

John Feo
john.feo@pnnl.gov
PNNL

Impact

Scalability:

- Expose, express and exploit new forms of parallelism
- Provide mechanisms of task scheduling across the system as if it were one system rather than many disparate pieces
- Symmetric access semantics across heterogeneous devices

Locality:

- Provide mechanisms to express locality as a first-class citizen
- Expose the memory hierarchies to the compiler (and programmer)
- Provide data types and memory models so that the programmer can view the system as one system instead of many disparate memories

Programmability:

- Create easier ways of expressing asynchrony thereby enabling programmers to write more scalable programs
- R-Stream will automatically extract parallelism and locality from common idioms
- Provide data types and algorithms that provide high-level representations of arrays mapped to the memory and algorithm hierarchy for automatic parallelization and data placement

Portability:

- Demonstrate a software stack that is portable to multiple architectures provided a C compiler
- Support a platform abstraction layer in SWARM, which will allow it to operate on multiple heterogeneous architectures
- Work with Xpress on the XPI interface to show application portability between runtime systems

Energy Efficiency:

- Collocate execution and data
- Dynamically load balance execution based on resource availability
- Dynamically scale resources based on load
- Provide new programming constructs (Rescinded Primitive Data Types) that allow compressed data formats at higher memory levels to minimize data transfer costs

Resilience:

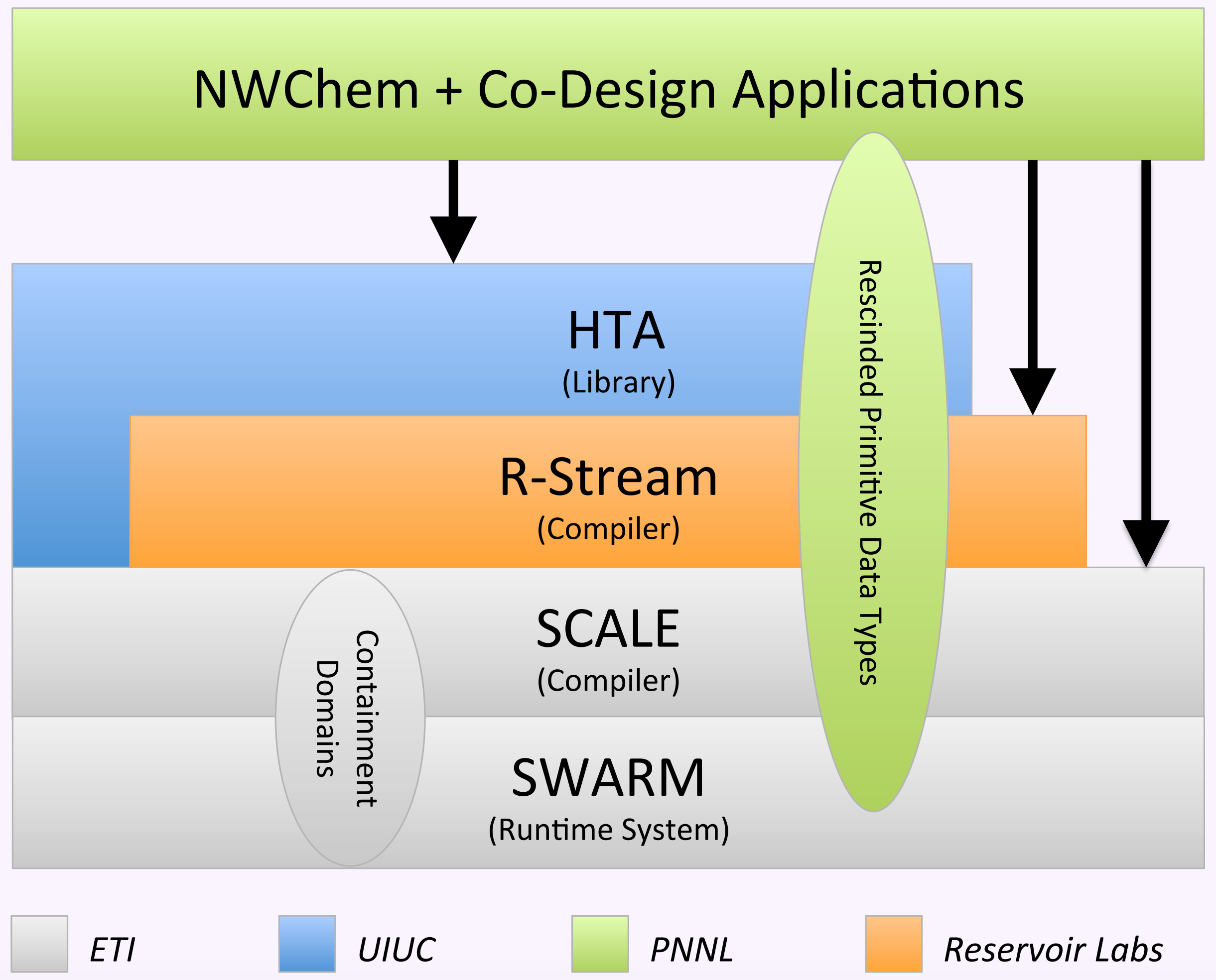
- Integrate containment domains and their extensions into the SWARM runtime system and SCALE compiler
- Allow graceful degradation in the face of exascale-level faults and a framework for software validation of soft faults

Interoperability:

- Work with Xpress on XPI interoperability with legacy codes such that all X-Stack runtime systems and all X-Stack applications can benefit from Evolutionary/Revolutionary runtime system interoperability

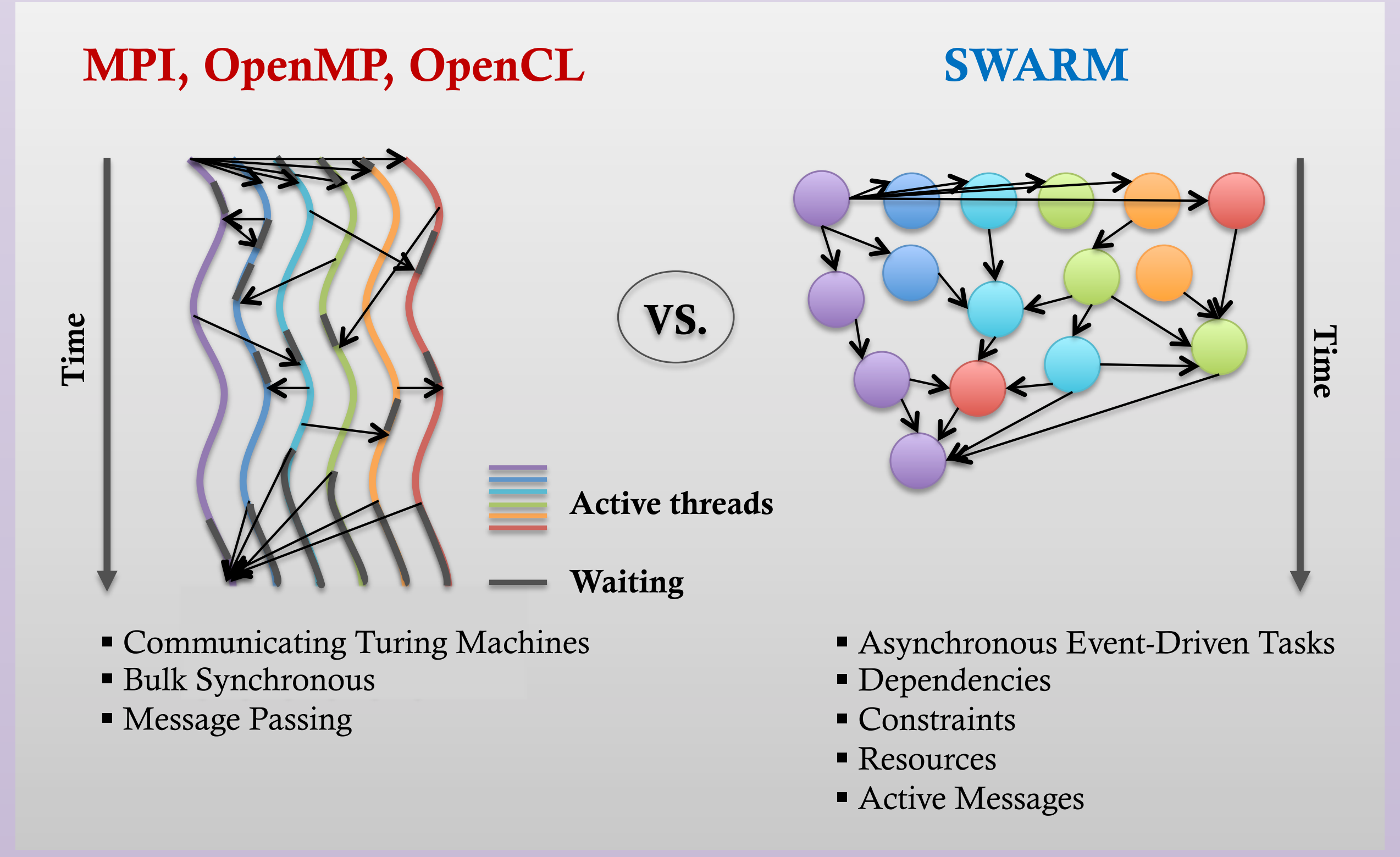
Applications:

- Provide NWChem kernels and expertise to all X-Stack projects
- Use Co-Design and NWChem applications to evaluate the Brandywine Team Software Stack



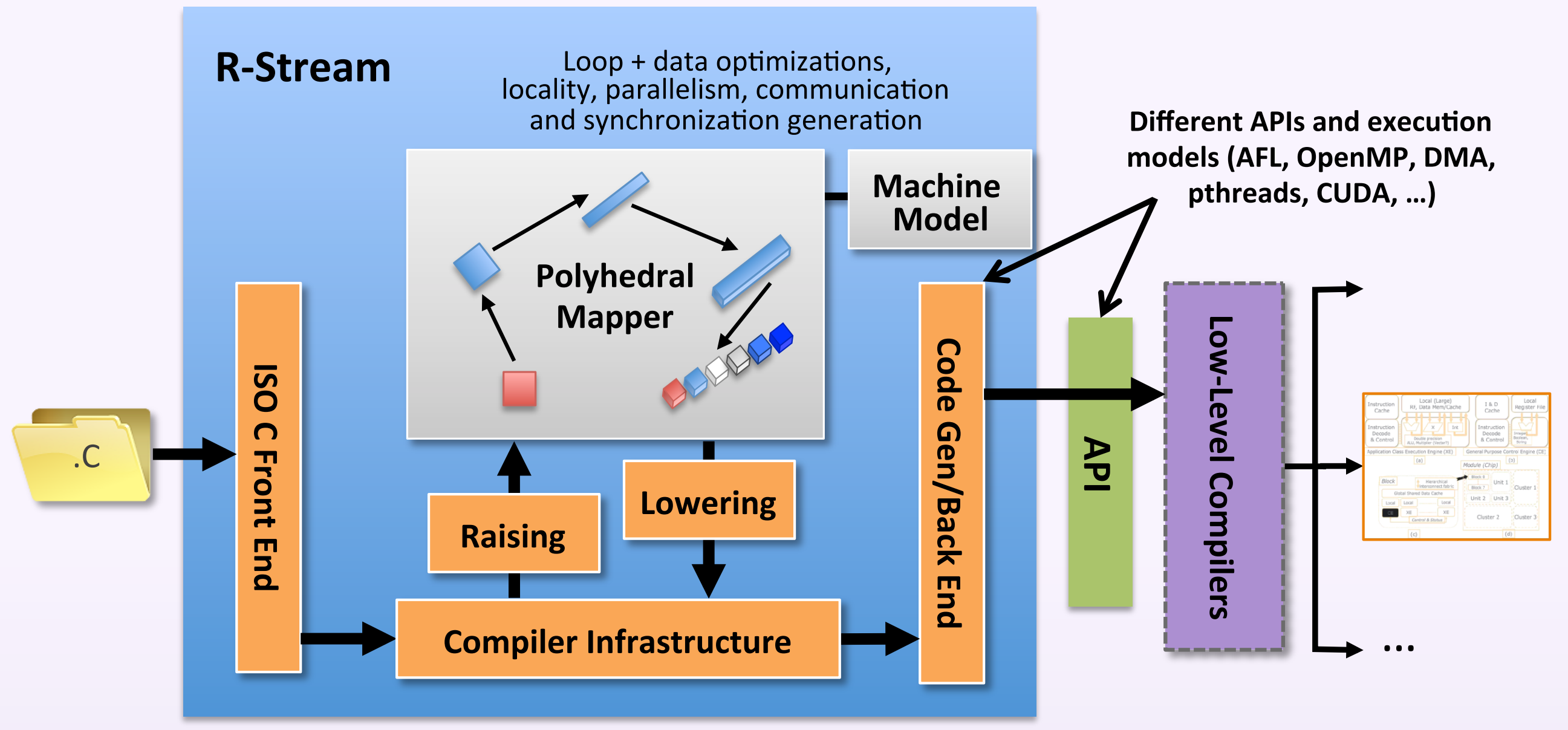
Brandywine Team Software Stack

The X-Stack software stack (figure above) consists of high level data objects and algorithms (HTA: Hierarchical Tiled Arrays), R-Stream loop optimizing compiler, SCALE parallel language compiler, and SWARM distributed heterogeneous runtime system. The project will extend the existing software tools to improve on parallelism, locality, programmability, portability, energy efficiency, resilience, and interoperability (see left). In addition, it will add new infrastructure for energy efficiency (Rescinded Primitive Data Types) and resilience (Containment Domains).



SWARM

- Codelets
  - Basic unit of parallelism
  - Nonblocking tasks
  - Scheduled upon satisfaction of precedent constraints
- Hierarchical Locale Tree: spatial position, data locality
- Lightweight Synchronization
- Asynchronous Split-phase Transactions: latency hiding
- Message Driven Computation
- Control-flow and Dataflow Futures
- Error Handling
- Active Global Address Space (planned)
- Fault tolerance (planned)



R-Stream
Loop + data optimizations, locality, parallelism, communication and synchronization generation
.C
ISO C Front End
Polyhedral Mapper
Machine Model
Raising
Lowering
Compiler Infrastructure
Code Gen/Back End
API
Low-Level Compilers
Different APIs and execution models (AFL, OpenMP, DMA, pthreads, CUDA, …)
…

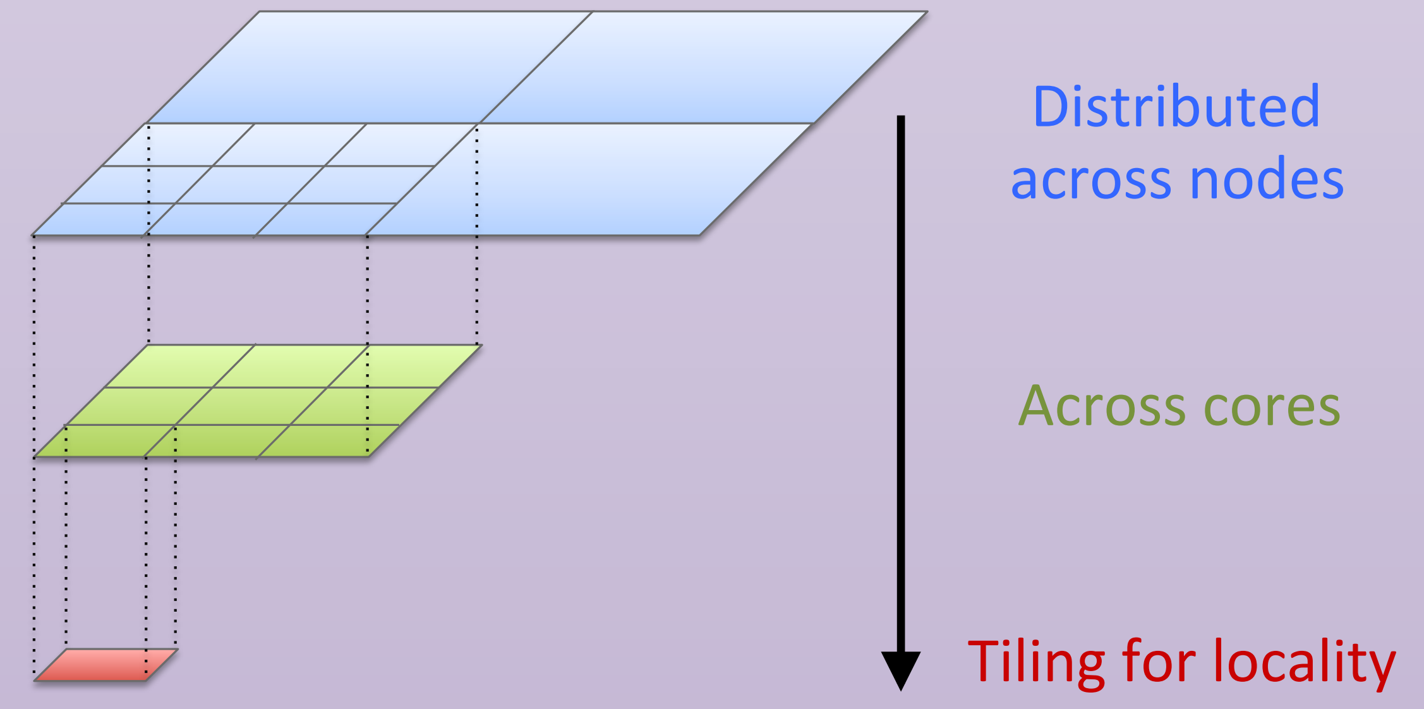
R-Stream

Current capabilities:

- Automatic parallelization and mapping
- Heterogeneous, hierarchical targets
- Automatic DMA/comm. generation/optimization
- Auto-tuning tile sizes, mapping strategies, etc.
- Scheduling with parallelism/locality layout tradeoffs
- Corrective array expansion

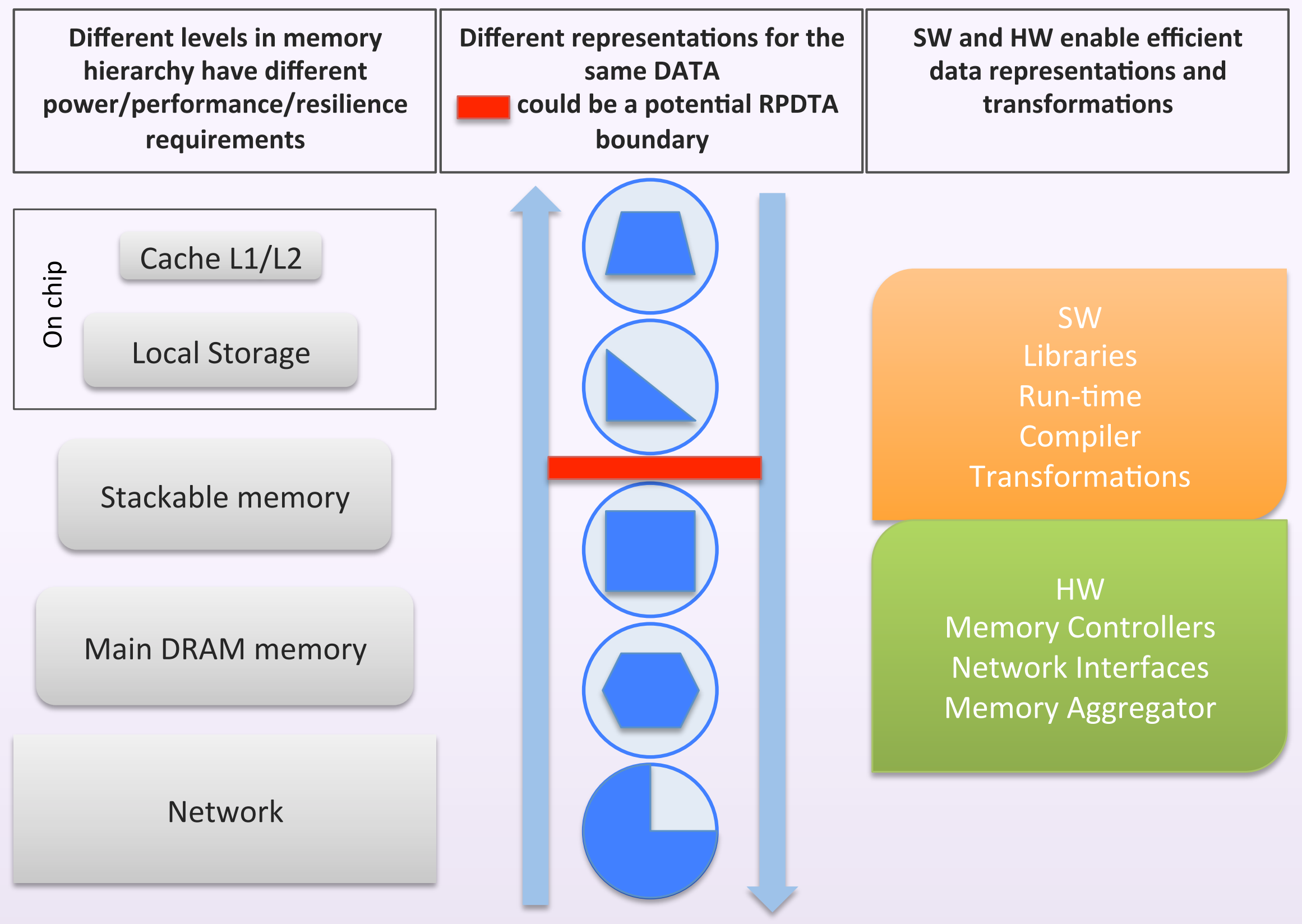
Planned capabilities:

- Extend explicit data placement
- Generation of parallel codelet codes from serial codes
- Generation of SCALE IR and tuning hints on scheduling and data placement
- Automatic mapping of irregular mesh codes



Hierarchical Tiled Arrays

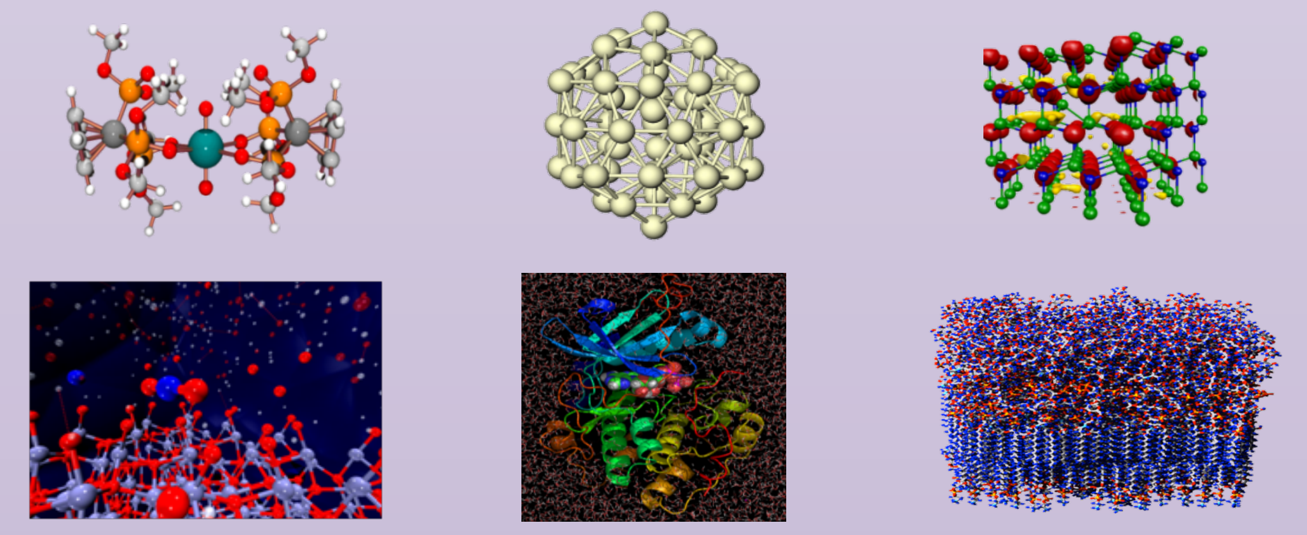
- HTAs are recursive data structure
  - Tree structured representation of memory
- Includes library of operations to enable the programming of codelets in the familiar notation of C/C++
  - Represent parallelism using operations on arrays and sets
  - Represent parallelism using parallel constructs such as parallel loops
- Compiler optimizations on sequences of HTA operations will be evaluated



Different levels in memory hierarchy have different power/performance/resilience requirements
Different representations for the same DATA
🟥 could be a potential RPDTA boundary
SW and HW enable efficient data representations and transformations
On chip
Cache L1/L2
Local Storage
Stackable memory
Main DRAM memory
Network
SW
Libraries
Run-time
Compiler
Transformations
HW
Memory Controllers
Network Interfaces
Memory Aggregator

Rescinded Primitive Data Type Access

- Redundancy removal to improve performance/energy
  - Communication
  - Storage
- Redundancy addition to improve fault tolerance
  - High Level fault tolerant error correction codes and their distributed placement
- Placeholder representation for aggregated data elements
  - Memory allocation/deallocation/copying
  - Memory consistency models



NWChem

- DOE's Premier computational chemistry software
- One-of-a-kind solution scalable with respect to scientific challenge and compute platforms
- From molecules and nanoparticles to solid state and biomolecular systems
- Open-source has greatly expanded user and developer base (ECL 2.0)
- Worldwide distribution (70% is academia)
- Ab initio molecular dynamics runs at petascale
- Scalability to 100,000 processors demonstrated
- Smart data distribution and communication algorithms enable hybrid-DFT to scale to large numbers of processors